Abstract

Several recent works have considered the problem of generating reviews (or 'tips') as a form of *explanation* as to why a recommendation might match a user's interests. While promising, we demonstrate that existing approaches struggle (in terms of both quality and content) to generate *justifications* that are relevant to users' decision-making process. We seek to introduce new datasets and methods to address this *recommendation justification* task. In terms of data, we first propose an 'extractive' approach to identify review segments which justify users' intentions; this approach is then used to distantly label massive review corpora and construct large-scale personalized recommendation justification datasets. In terms of generation, we design two personalized generation models with this data: (1) a reference-based Seq2Seq model with aspect-planning which can generate justifications covering different aspects, and (2) an aspect-conditional masked language model which can generate diverse justifications based on templates extracted from justification histories. We conduct experiments on two real-world datasets which show that our model is capable of generating convincing and diverse justifications.

1 Introduction

Explaining, or justifying, recommendations to users has the potential to increase their transparency and reliability. However providing meaningful interpretations remains a difficult task, partly due to the black-box nature of many recommendation models, but also because we simply lack ground-truth datasets specifying what 'good' justifications ought to look like.

Previous work has sought to learn user preferences and writing styles from crowd-sourced reviews (Dong et al., 2017; Ni and McAuley, 2018) to generate explanations in the form of natural language, e.g. generating synthesized reviews similar to those that users would write about a product. However, a large portion of review text (or text from 'tips') is often of little relevance to most users' decision making (e.g. they describe verbose experiences or general endorsements) and may not be appropriate to use as explanations in terms of content and language style. As a result, existing models that learn directly from reviews (or tips) may not capture crucial information that explains users' purchases. Table 1 shows examples of reviews, tips and ideal justifications. More recently, there has been work studying the task of tip generation where tips are concise summaries of reviews (Li et al., 2017). Though tips are concise and some subset of them might be suitable as candidates for recommendation justifications, only a few e-commerce systems provide tips accompanied with reviews. Even in systems where tips are available, the number of tips is usually far smaller than the number of reviews. These approaches hence suffer from generalizability issues, especially in settings

Review examples:
I love this little stand! **The coconut mocha chiller and caramel macchiato are delicious.**
Wow what a special find. One of the most unique and special date nights my husband and I have had.
Tip examples:
Great food. Nice ambiance. Gnocchi were very good.
I can't get enough of this place.
Justification examples:
The food portions were huge.
Plain cheese quesadilla is very good and very cheap.

Table 1: In contrast to reviews and tips, we seek to automatically generate *recommendation justifications* that are more concise, concrete, and helpful for decision making. Examples of justifications from reviews, tips, and our annotated dataset are marked in bold.

where user interactions are highly sparse.

On the other hand, generating *diverse* responses is essential in personalized content generation scenarios such as justification generation. Instead of always predicting the most popular reasons, it's preferable to present diverse justifications for different users based on their personal interests. Recent work has shown that incorporating prior knowledge into generation frameworks can greatly improve diversity. Prior knowledge could include story-lines in story generation (Yao et al., 2019), or historical responses in dialogue systems (Weston et al., 2018).

In this work, our goal is to generate convincing and diverse justifications. To address the challenge of lacking ground-truth data about 'good' justifications, we propose a pipeline that can identify justifications from massive corpora of reviews or tips. We extract fine-grained aspects from justifications and build user personas and item profiles consisting of sets of representative aspects. To improve generation quality and diversity, we propose two generation models (1) a reference-based Seq2Seq model with aspect-planning, which takes previous justifications as a reference and can produce justifications based on different aspects, and (2) an aspect-conditional masked language model that can generate diverse justifications from templates extracted from previous justifications.

Our contributions are threefold:

- To facilitate recommendation justification generation, we propose a pipeline to identify justification candidates and build aspect-based user personas and item profiles from massive corpora of reviews. With this approach, we are able to build large-scale personalized justification datasets. We use these extractive justification segments in the task of explainable recommendation and show that these are better training sources instead of whole reviews.
- We propose two models based on reference attention, aspect-planning techniques and a persona-conditional masked language model. We show that adding such personalized information enables the models to generate justifications with high quality and diversity.
- We conduct extensive experiments on two real-world datasets from *Yelp* and *Amazon Clothing*. We provide an annotated dataset about 'good' justifications on the Yelp dataset and show that the binary classifier trained on this dataset generalizes well to the Amazon Clothing dataset. We study different decoding strategies and compare their effect on generation performance.

2 Dataset Generation

In this section, we introduce the pipeline to extract high quality justifications from raw user reviews. Specifically, our goal here is to identify review segments that can be used as justifications and build a personalized justification dataset upon them. Our pipeline consists of three steps:

1. Annotating a set of review segments with binary labels, *i.e.,* to determine whether they are 'good' or 'bad' justifications.
2. Training a classifier on the annotated subset and applying it to distantly label all the review segments to extract 'good' justifications for each user and item pair.
3. Applying fine-grained aspect extraction for the extracted justifications, and building user personas and item profiles.

2.1 Identifying Justifications From Reviews

The first step is to extract text segments from reviews that are appropriate to use as justifications. Instead of a complete sentence or phrase, we define each segment as an Elementary Discourse Unit (EDU; (Mann and Thompson, 1988)) which corresponds to a sequence of clauses. We use the model of Wang et al. (2018) to obtain EDUs from reviews. Recent works have shown that EDUs can improve the performance of document-level summarization (Bhatia et al., 2015) and opinion summarizaiton (Angelidis and Lapata, 2018).

After preprocessing the reviews into EDUs, we analyzed the linguistic differences between recommendation justifications and reviews, and built two rules to filter the segments that are unlikely to be suitable justifications: (1) segments with first-person or third-person pronouns, and (2) too long or short. Next, two expert annotators were exposed to 1,000 segments among those not filtered out and asked to determine whether they are 'good' justifications. Labeling was performed iteratively, followed by feedback and discussion, until the quality was aligned between the two annotators. At the end of the process, the inter-annotator agreement for the binary labeling task (*good* vs. *bad*), measured by Cohen's kappa (Cohen, 1960), was 0.927 after alignment. Then, the annotators further labeled 600 segments. Overall,

Method	F1	Recall	Precision
BOW-Xgboost	0.559	0.679	0.475
CNN	0.644	0.596	0.700
LSTM-MaxPool	0.675	0.703	0.650
BERT	**0.747**	0.700	**0.800**
BERT-SA (one epoch)	0.481	0.975	0.320
BERT-SA (three epoch)	0.491	**1.000**	0.325

Table 2: Performance for classifying review segments as good or bad for recommendation justification.

24.8% of the segments were labeled *good*.

2.2 Automatic Classification

Our next step is to propagate labels to the complete review corpus. Here we adopt BERT (Devlin et al., 2019) to fine-tune on our classification task, where a [CLS] token is added to the beginning of each segment and the final hidden state (i.e., output of BERT) corresponding to this token is fed into a linear layer to obtain the binary prediction. Cross entropy is used as the training loss.

We split the annotated dataset into Train, Dev, and Test sets with a 0.8/0.1/0.1 ratio, fine-tune the BERT classifier on the Train set and choose the best model on the Dev set. After three epochs of fine-tuning, BERT can achieve an F1-score of 0.80 on the Test set. We compare the performance of BERT with multiple baseline models: (1) a XGBoost model which uses Bags-of-Words as sentence features (2) a convolutional neural network (CNN) with three convolution layers and one linear layer (3) a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) network with a max-pooling layer, and a linear layer (4) a BERT sentiment classifier (BERT-SA) trained on the complete Yelp dataset for one epoch and three epochs. To obtain the pre-trained word embeddings for the CNN and LSTM models, we applied fastText (Bojanowski et al., 2016) on the *Yelp* Review dataset. We set the embedding dimension to 200 and used default values for other hyper-parameters.

Table 2 presents results for our binary classification task. The BERT classifier has higher F1-score and precision than other classifiers. The BERT-SA model after three epochs only achieves an F1-score of 0.491, which confirms the difference between sentiment analysis and our *good/bad* task, i.e., even if the segment has positive sentiment, it might be not suitable as a justification.

Yelp
The *Tuna* is pretty amazing
Appetizers and *pasta* are excellent here
An excellent *selection* of both *sweet* and savory *crepes*
It was filled with delicious *food*, fantastic *music* and *dancing*
Amazon-Clothing
The *quality* of the *material* is great
Great *shirt*, especially for the *price*.
The *seams* and *stitching* are really nice
Fit the bill for a *Halloween* *costume*.

Table 3: Examples of justifications with fine-grained aspects in our annotated dataset. The fine-grained aspects are italic and underlined.

2.3 Fine-grained Aspect Extraction

Finally, we extract the fine-grained aspects that each justification covers. Fine-grained aspects are properties of products that appear among a user's opinions. We adopt the method proposed by Zhang et al. (2014) to build a sentiment lexicon which includes a set of fine-grained aspects from the whole dataset. We then use simple rules to determine which aspects appear in each justification.[1] Table 3 presents a set of examples from our dataset. Each example consists of a justification that a user has written about an item, and multiple fine-grained aspects mentioned in the justification. **Note that** we only annotated the Yelp dataset, trained a classifer on that and applied the model on both Yelp and Amazon Clothing dataset. As shown in Table 3, the trained classifier works well on both datasets.

3 Approach

3.1 Problem Definition

For each user u (or item i), we build a justification reference $D = \{d_1, \dots, d_{l_r}\}$ consisting of justifications that the user has written (or justifications about the item) on the training set, where l_r is the maximum number of justifications. We also obtain a user persona (or item profile) $A = \{a_1, \dots, a_K\}$ based on the fine-grained aspects that the user's (or item's) previous justifications have covered, where K is the maximum number of aspects.

Given a user u and an item i, as well as the their justification reference D_u and D_i, and u's persona A_u and i's profile A_i, **our target** is to predict the

[1]For each aspect, if its singular or plural exists in the tokenized justification, then we consider that this aspect exists in that justification.

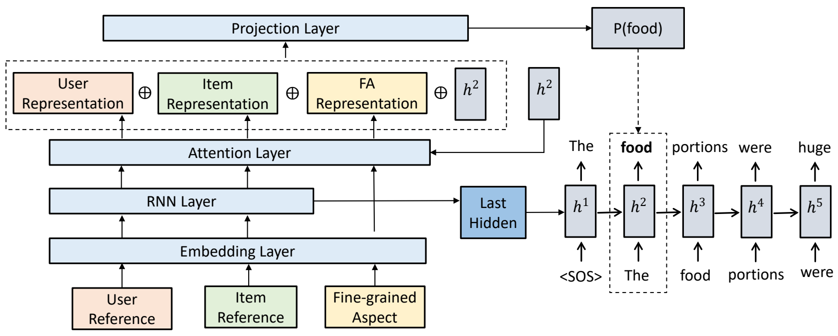

Figure 1: Structure of the reference-based Seq2Seq model with Aspect Planning

justifications $J_{u,i} = \{w_1, w_2, \dots, w_T\}$ that would explain why item i fits user u's interests, where T is the length of the justification.

3.2 Reference-based Seq2Seq Model

Our base model follows the structure of a standard Seq2Seq (Sutskever et al., 2014) model. Our framework, called 'Ref2Seq', views the historical justifications of users and items as references and learns latent personalized features from them. Figure 1 shows the structure of our Reference-based Seq2Seq Model. It includes two components: (1) two sequence encoders that learn user and item latent representations by taking previous justifications as references; (2) a sequence decoder incorporating representations from users and items to generate personalized justifications.

Sequence Encoders. Our user encoder and sequence encoder share the same structure, which includes an embedding layer, a two-layer bi-directional GRU (Cho et al., 2014), and a projection layer. The inputs are a user (or item) reference D consisting of a set of historical justifications. These justifications pass a word embedding layer, then go through the GRU and yield a sequence of hidden states $\mathbf{e} \in \mathbb{R}^{l_s \times l_r \times n}$:

$$\mathrm{E} = \mathrm{Embedding}(D), \mathbf{e} = \mathrm{GRU}(\mathrm{E}) = \overrightarrow{\mathrm{e}} + \overleftarrow{\mathrm{e}}, \tag{1}$$

where l_s denotes the length of the sequence, n is the hidden size of the encoder GRU, $\mathrm{E} \in \mathbb{R}^{l_s \times l_r \times n}$ is the embedded sequence representation, and $\overrightarrow{e}$ and $\overleftarrow{e}$ are the hidden vectors produced by a forward and a backward GRU (respectively).

To combine information from different 'references' (i.e. justifications), the hidden states are then projected via a linear layer:

$$\hat{\mathbf{e}} = W_e \cdot \mathbf{e} + \mathbf{b}_e, \tag{2}$$

where $\hat{\mathbf{e}} \in \mathbb{R}^{l_s \times n}$ is the final output of the encoder, and $W_e \in \mathbb{R}^{l_r}$, $\mathbf{b}_e \in \mathbb{R}$ are learned parameters.

Sequence decoder. The decoder is a two-layer GRU that predicts the target words given a start token. The hidden state of the decoder is initialized using the sum of the last hidden state of the user and item encoders. The hidden state at time-step t is updated via the GRU unit based on the previous hidden state and the input word. Specifically:

$$\mathbf{h}_0 = \mathbf{e}^u_{l_s} + \mathbf{e}^i_{l_s}, \mathbf{h}_t = \mathbf{GRU}(w_t, \mathbf{h}_{t-1}), \tag{3}$$

where $\mathbf{e}^u_{l_s}$ and $\mathbf{e}^i_{l_s}$ are the last hidden states of the user and item encoder output $\hat{\mathbf{e}}_u$ and $\hat{\mathbf{e}}_i$.

To explore the relation between the reference and generation, we apply an attention fusion layer to summarize the output of each encoder. For the user and item reference encoder, the attention vector is defined as:

$$\mathbf{a}^1_t = \sum_{j=1}^{l_s} \alpha^1_{tj} \mathbf{e}_j,$$
$$\alpha^1_{tj} = \exp(\tanh(\mathbf{v}^{1\,\top}_{\alpha}(W^1_{\alpha}[\mathbf{e}_j; \mathbf{h}_t] + \mathbf{b}^1_{\alpha})))/Z, \tag{4}$$

where $\mathbf{a}^1_t \in \mathbb{R}^n$ is an attention vector on the sequence encoder at time-step t, α^1_{tj} is an attention score over the encoder hidden state $\mathbf{e}_j$ and decoder hidden state $\mathbf{h}_t$, and Z is a normalization term.

Aspect-Planning Generation. One of the challenges for generating justifications is how to improve controllability, i.e., directly manipulate the content being generated. Inspired by 'plan-and-write' (Yao et al., 2019), we extend the base model to an Aspect-Planning Ref2Seq (AP-Ref2Seq) model where we plan a fine-grained aspect before generation. This aspect planning can be considered as an extra form of supervision instead of a hard constraint to make justification generation more controllable.

When generating the justification for user u and item i, we first provide a fine-grained aspect a as a plan. The aspect a is fed into the word embedding layer to obtain the aspect embedding $\mathrm{E_a}$. Then, we compute the scores between the embedding of the aspect and the decoder hidden state as:

$$\begin{aligned}\mathbf{a}_t^2 &= \alpha_t^2 \mathrm{E}_a,\\ \alpha_t^2 &= \exp(\tanh({\mathbf{v}_\alpha^2}^\top(W_\alpha^2[\mathrm{E}_a; \mathbf{h}_t] + \mathbf{b}_\alpha^2)))/Z,\end{aligned} \tag{5}$$

where $\mathbf{a}_t^2 \in \mathbb{R}^n$ is an attention vector and α_t^2 is an attention score.

The attention vectors $\mathbf{a}_{ut}^1$ of user u, $\mathbf{a}_{it}^1$ of item i, and $\mathbf{a}_t^2$ of fine-grained aspect a, are concatenated with the decoder hidden state at time-step t and projected to obtain the output word distribution P. The output probability for word w at time-step t is given by:

$$p(w_t) = \tanh(W_1[\mathbf{h}_t; \mathbf{a}_{ut}^1; \mathbf{a}_{it}^1; \mathbf{a}_t^2] + \mathbf{b}_1), \tag{6}$$

where w_t is the target word at time-step t. Given the probability $p(w_t)$ at each time step t, the model is trained using a cross-entropy loss compared against the ground-truth sequence.

3.3 Aspect Conditional Masked Language Model

Though Seq2Seq-based models can achieve high quality output, they often fail to generate *diverse* content. Recent works in natural language generation (NLG) tried to combine generation methods with information retrieval techniques to increase the generation diversity (Li et al., 2018; Baheti et al., 2018). The basic idea follows the paradigm of retrieve-and-edit—which is to first retrieve historical responses as templates, and then edit the template into new content. Since our data is annotated with fine-grained aspects, it naturally fits into this type of retrieve-and-edit paradigm. Meanwhile, masked language models have shown great performance in language modeling. Recent work (Wang and Cho, 2019; Mansimov et al., 2019) has shown that by sampling from the masked language model (e.g. BERT), it is able to generate coherent sentences.

Inspired by this work, we want to extend such an approach into a conditional version—we explore the use of an Aspect Conditional Masked Language Model (ACMLM) to generate diverse personalized justifications. Figure 2 shows the structure of our Aspect Conditional Masked Language Model. For a justification $J_{u,i}$ that user u wrote about item i, we adapt the pre-trained BERT model (Devlin et al., 2019) into an encoder-decoder network with (1) an aspect encoder which encodes the user persona and item profile into latent representations and (2) a masked language model sequence decoder that takes in a masked justification and predicts the masked tokens.

Aspect Encoder. Our aspect encoder shares the same WordPiece embeddings (Wu et al., 2016) as BERT. The encoder feeds the intersection of fine-grained aspects from the user persona and item profile $A_{ui} = \{a_1, \ldots, a_{K'}\}$ into the embedding layer and obtains the aspect embedding $\mathbf{A}_{ui} \in \mathbb{R}^{K' \times n}$, where K' is the number of common fine-grained aspects and n is the dimension of the WordPiece embeddings.

Masked Language Model Sequence Decoder. We use the masked language model in the pre-trained BERT model as our sequence decoder and add attention over the aspect encoder's output. As shown in Figure 2, the input to the decoder is a masked justification $J_{u,i}^M = \{w_1, \ldots, w_T\}$ with multiple tokens be replaced as [MASK]. The decoder's output $\mathbf{T} \in \mathbb{R}^{T \times n}$ is then fed to the attention layer to calculate an attention score with the output of the encoder:

$$\begin{aligned}\mathbf{a}_t^3 &= \sum_{j=1}^{K'} \alpha_{tj}^3 \mathbf{A}_j,\\ \alpha_{tj}^3 &= \exp(\tanh({\mathbf{v}_\alpha^3}^\top(W_\alpha^3[\mathbf{A}_j; \mathbf{T}_t] + \mathbf{b}_\alpha^3)))/Z.\end{aligned} \tag{7}$$

The attention vector $\mathbf{a}_t^3$ is then concatenated with the decoder hidden state at time-step t and sent to a linear projection layer to obtain the output word distribution P. The output probability for word w at time-step t is given by:

$$p(w_t) = \tanh(W_2[\mathbf{T}_t; \mathbf{a}_t^3] + \mathbf{b}_2) \tag{8}$$

where w_t is the target word at time-step t.

Masking Procedure. The original BERT paper applies a flat rate (15%) to decide whether to mask a token. Unlike their approach, we adopt a higher rate to mask fine-grained aspects since they are more important in justifications. Specifically, if we encounter a fine-grained aspect, we will replace it with a [MASK] token 30% of the time; while for other words, we will replace them with a [MASK] token 15% of the time.

During training, the model will only predict those masked tokens and calculate a cross-entropy loss on them.

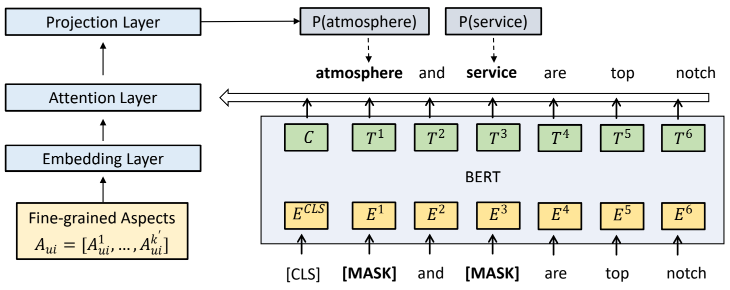
Projection Layer
P(atmosphere)
P(service)
atmosphere and service are top notch
Attention Layer
Embedding Layer
Fine-grained Aspects $A_{ui} = [A^1_{ui}, ..., A^{k'}_{ui}]$
C T^1 T^2 T^3 T^4 T^5 T^6
BERT
E^{CLS} E^1 E^2 E^3 E^4 E^5 E^6
[CLS] [MASK] and [MASK] are top notch

Figure 2: Structure of the Aspect Conditional Masked Language Model

Iter 0	universe [MASK] is extremely friendly and persona ##ble
Iter 5	the [MASK] is extremely friendly and persona ##ble
Iter 10	the [MASK] is extremely friendly and persona ##ble
Iter 15	the staff are extremely cool and persona ##ble
Iter 20	the staff are extra kind , persona ##ble

Table 4: Examples of the generation output of ACMLM at different iterations.

Generation by Sampling from Masked Templates. We next discuss how to generate justifications from the trained ACMLM. We follow the sampling strategy of Wang and Cho (2019) to generate justifications. Instead of generating from a sequence of all [MASK] tokens, we start with masked templates generated from historical justifications about the target item. These masked templates include prior knowledge about the item and can increase the speed of sampling convergence.

Table 4 shows an example of the generation process. We initialize the template sequence X^0 as (universe, [MASK], $\dots$, ##ble) with length T. At each iteration i, a position t_i is sampled uniformly at random from $\{1, \dots, T\}$ and the token at t_i (i.e. $x^i_{t_i}$) of the current sequence X^i is replaced by [MASK]. After that, we obtain the conditional probability of x_{t_i} as

$$p(x_{t_i}|X^i_{\setminus t_i} = \frac{1}{Z(X^i_{\setminus t_i})} \exp(1\text{h}(x_{t_i})^\top f_\theta(X^i_{\setminus t_i}))), \tag{9}$$

where $1\text{h}(x_{t_i})$ is a one-hot vector with index x_{t_i} set to 1, $X^i_{\setminus t_i}$ is the sequence we obtain after replacing the token at position t_i of X^i by [MASK], $f_\theta(X^i_{\setminus t_i})$ is the output after feeding $X^i_{\setminus t_i}$ into the ACMLM as in Equation (8), and Z is the normalization term. We then sample $\tilde{x}_{ti}$ from Equation (9), and construct the next sequence by $X^{i+1} = (x^i_1, \dots, \tilde{x}_{ti}, \dots, x^i_T)$. After repeating this procedure N times, the final output is considered as the generation output.[2]

4 Experiments

4.1 Datasets

With our proposed pipeline (Section 2), we construct two personalized justification datasets from existing review data—Yelp and Amazon Clothing.[3][4] We further filter those users with fewer than five justifications. For each user, we randomly hold out two samples from all of their justifications to construct the Dev and Test sets. Table 5 shows the statistics of our two datasets.

4.2 Baselines

For automatic evaluation, we consider three baselines: Item-Rand is a baseline which randomly chooses a justification from the item's historical justifications. LexRank is a strong unsupervised baseline that is widely used in text summarization (Erkan and Radev, 2004). Given all historical justifications about an item, LexRank can select one justification as the summary. We then use that as the justification for all users. Attr2Seq (Dong et al., 2017) is a Seq2Seq baseline that uses attributes (i.e. user and item identity) as input.

By default, all models use beam search during generation. Recently, there have been works showing that the generation output of sampling methods is more diverse and suitable on high-entropy tasks (Holtzman et al., 2019). To this end, we explore another decoding strategy— 'Top-k sampling' (Radford et al., 2019) in experiments

[2]We set N proportional to the length T of the initial masked template to prevent the generation diverging too much from the original template.

[3]https://www.yelp.com/dataset/challenge

[4]http://jmcauley.ucsd.edu/data/amazon

Dataset	Train	Dev	Test	# Users	# Items	# Aspects
Yelp	1,219,962	115,907	115,907	115,907	51,948	2,041
Amazon Clothing	202,528	57,947	57,947	57,947	50,240	581

Table 5: Statistics of our datasets.

Dataset	Yelp				Amazon Clothing			
Model	BLEU-3	BLEU-4	Distinct-1	Distinct-2	BLEU-3	BLEU-4	Distinct-1	Distinct-2
Item-Rand	0.440	0.150	2.766	20.151	1.620	0.680	2.400	11.853
LexRank	2.290	0.920	1.738	8.509	3.480	2.250	2.407	14.956
Attr2seq	**7.890**	0.000	0.049	0.095	1.720	0.560	0.076	0.352
Ref2Seq	4.380	**2.450**	0.188	1.163	8.780	5.670	0.141	1.240
AP-Ref2Seq	3.390	1.830	0.326	2.094	**13.910**	**12.500**	0.557	3.661
Ref2Seq (Top-k)	1.630	0.700	0.818	11.927	3.960	2.130	0.697	**10.858**
ACMLM	0.700	0.280	**1.322**	**14.319**	2.420	1.590	**0.942**	9.312

Table 6: Performance on Automatic Evaluation.

and include a variant of our model: Ref2Ref (Top-k).[5]

For human evaluation, we include two baselines: Ref2Seq (Review) and Ref2Seq (Tip), both of which are the same model as Ref2Seq model but trained on the original review and tip data, respectively. Comparisons with these two baselines demonstrates that training on our annotated dataset tends to generate text more suitable as justifications.

Model	R	I	D
Ref2Seq (Review)	3.02	2.39	2.10
Ref2Seq (Tip)	3.25	2.35	2.34
Ref2Seq	3.87	3.13	2.96
Ref2Seq (Top-k)	**3.95**	**3.34**	3.39
ACMLM	3.23	3.29	**3.42**

Table 7: Performance on Human Evaluation, where R,I,D represents Relevance, Informativeness and Diversity, respectively.

4.3 Implementation Detail

We use PyTorch[6] to implement our models.

For Req2Seq and AP-Ref2Seq, we set the hidden size and word embedding size as 256. We apply a dropout rate of 0.5 for the encoder and 0.2 for the decoder. The size of the justification reference l_r is set to 5 and the number of fine-grained aspects K in the user persona and item profile is set to 30. We train the model using Adam with learning rate $2e^{-4}$ and stop training either when it reaches 20 epochs or the perplexity does not improve (on the Dev set). For ACMLM, we build our model based on the BERT implementation from HuggingFace.[7] We initialize our decoder using the pre-trained 'Bert-base' model and set the max sequence length to 30. We train the model for 5 epochs using Adam with learning rate $2e^{-5}$. For models using beam search, we set the beam size as 10. For models using 'top-k' sampling, we set k to 5. For ACMLM, we use a burn-in step equal to the length of the initial sequence. Our data and code are available online.[8]

[5]At each time step, the next word is sampled from the top k possible next tokens, according to their probabilities.

[6]http://pytorch.org/docs/master/index.html

[7]https://github.com/huggingface/pytorch-pretrained-BERT

4.4 Automatic Evaluation

For automatic evaluation, we use BLEU, Distinct-1, and Distinct-2 (Li et al., 2015) to measure the performance of our model. As shown in Table 6, our reference-based models achieve the highest BLEU scores on both datasets except for BLEU-3 on Yelp. This confirms that Ref2Seq is able to capture user and item content to generate the most relevant content, compared with unpersonalized models such as LexRank and personalized models that do not leverage historical justifications such as Attr2Seq.

On the other hand, recent works have reported that models achieving higher diversity scores will have lower scores on overlap-based metrics (e.g. BLEU) for open-domain generation tasks (Baheti et al., 2018; Gao et al., 2018). We make a similar observation for our personalized justifi-

[8]https://github.com/nijianmo/recsys_justification.git

Model	Shake Shack	Teharu Sushi	MGM Grand Hotel
Ground Truth	The burger was good	The rolls are pretty great , typical rolls not that many specials	Room was very clean comfortable
LexRank	A great burger and fries.	Sushi ?	Great rooms.
Ref2Seq (Review)	i love trader joe 's , i love trader joe 's	the food was good and the service was great	i love this place ! the food is always good and the service is always great
Ref2Seq (Tip)	this place is awesome	love this place	come here
Ref2Seq	this place has some of the best burgers	the sushi is delicious	the room was nice
Ref2Seq (Top-k)	the fries are amazing	fresh and delicious sushi	open hotel for hours
ACMLM	breakfast sandwiches are overall very filling	overall fun experience with half price sushi	family style dinner , long time shopping trip to vegas, family dining , cheap lunch

Table 8: Comparisons of the generated justifications from different models for three businesses on the Yelp dataset.

cation generation task. As shown in Table 6, both sampling-based methods Ref2Seq (Top-k) and ACMLM achieve higher Distinct-1 and Distinct-2, while their BLEU scores are lower than Seq2Seq based models using beam search. Therefore, we also perform human evaluation to validate the generation quality of our proposed methods.

4.5 Human Evaluation

We conduct human evaluation on three aspects: (1) *Relevance* measures whether the generated output contains information relevant to an item; (2) *Informativeness* measures whether the generated justification includes specific information that is helpful to users; and (3) *Diversity* measures how distinct the generated output is compared with other justifications.

We focus on the Yelp dataset and sample 100 generated examples from each of the five models as shown in Table 7. Human annotators are asked to give a score in the range [1,5] (lowest to highest) for each metric. Each example is rated by at least three annotators. The results show that both Ref2Seq (Top-k) and ACMLM achieve higher scores on Diversity and Informativeness compared to other models.

4.6 Qualitative Analysis

Here we study the following two qualitative questions:

RQ1: How do training data and methods affect generation? As Table 8 shows, models trained on reviews and tips tend to generate generic phrases (such as ‘i love this place’) which often do not include information that helps users to make decisions. Other models trained on the justification datasets tend to mention concrete information (e.g. different aspects). LexRank tends to generate relevant but short content. Meanwhile, sampling-based models are able to generate more diverse content.

Dataset	Aspects	Generated Output
Yelp	dining	the dining room is nice
	pastry	the pastries were pretty good
	chicken	the chicken fried rice is the best
	sandwich	the pulled pork sandwich is the best thing on the menu
Amazon-Clothing	product	great product , fast shippong
	price	design is nice , good price
	leather	comfortable leather sneakers . classic
	walking	sturdy , great city walking shoes

Table 9: Generated justifications from AP-Ref2Seq. The planned aspects are randomly selected from users’ personas.

RQ2: How does aspect planning affect generation? To mitigate the trade-off between diversity and relevance, one approach is to add more constraints during generation such as constrained Beam Search (Anderson et al., 2017). In our work, we extend our base model Ref2Seq by incorporating aspect-planning to guide generation. As shown in Table 9, most planned aspects are present in the generated outputs of AP-Req2Seq.

5 Related Work

Explainable Recommendation There has been a line of work that studies how to improve the explainability of recommender systems. Catherine and Cohen (2017) learn latent representations of review text to predict ratings. These representations are then used to find the most helpful reviews for given a particular user and item pair. Another popular direction is to generate text to justify recommendations. Dong et al. (2017) proposed an attribute-to-sequence model to generate product reviews which utilizes categorical attributes. Ni et al. (2017) developed a multi-task learning method that considers collaborative filter and review generation. Li et al. (2019b) generated tips by considering 'persona' information which can capture the language style of users and characteristics of items. However, these works use whole reviews or tips as training examples, which may not be appropriate due to the quality of review text. More recently, Liu et al. (2019) proposed a framework to generate fine-grained explanations for text classification. To achieve labels for human-readable explanations, they constructed a dataset from a website which provides ratings and fine-grained summaries written by users. Unfortunately, most websites do not provide such fine-grained information. On the other hand, our work identifies justifications from reviews, uses them as training examples and shows these are better data source for explainable recommendation via extensive experiments.

Diversity-aware NLG Diversity is an important aspect of NLG systems. Recent works have focused on digesting prior knowledge to improve generation diversity. Yao et al. (2019) proposed a method to incorporate planned story-lines in story generation. Li et al. (2019a) developed an aspect-aware coarse-to-fine review generation method. They predict an aspect for each sentence in the review to capture the content flow. Given the aspects, a sequence of sentence sketches is generated and a decoder will fill in the slots of each sketch. In dialogue systems, several works have studied frameworks to extract templates from historical responses, which are then edited to form new responses (Weston et al., 2018; Wu et al., 2018). Similarly, the extract-and-edit paradigm has been studied in style transfer tasks in NLG (Li et al., 2018). Wu et al. (2019) proposed an attribute aware masked language model for non-parallel sentiment transfer. They first mask out the sentimental tokens and then train a masked language model to infill the masked positions for target sentiment. In this work, we also introduces a conditional masked language model but considers more fine-grained aspects.

6 Conclusion

In this work, we studied the problem of personalized justification generation. To build high quality justification datasets, we provided an annotated dataset and proposed a pipeline to extract justifications from massive review corpora. To generate convincing and diverse justifications, we developed two models: (1) Ref2Seq which leverages historical justifications as references during generation; and (2) ACMLM, which is an aspect conditional model built on a pre-trained masked language model. Our experiments showed that Ref2Seq achieves higher scores (in terms of BLEU) and ACMLM achieves higher diversity scores compared with baselines. Human evaluation showed that reference-based models obtain high relevance scores and sampling based methods led to more diverse and informative outputs. Finally, we showed that aspect-planning is a promising way to guide generation to produce personlized and relevant justifications.

Acknowledgements. This work is partly supported by NSF #1750063. We thank all the reviewers for their constructive suggestions.

References

Peter Anderson, Basura Fernando, Mark Johnson, and